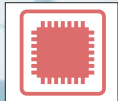
基礎から学ぶ Verilog HDL & FPGA 設計

第１回

全加算器を HDL で設計してみよう

中野浩嗣，伊藤靖朗

デバイスの記事　ビギナーズ　関連データ

Verilog HDLによるFPGA（field programmable gate array）設計を基礎から学ぶための連載記事である．今回は，最も単純な組み合わせ回路の一つである全加算器を設計する．（編集部）

この連載では，シミュレーションやFPGAボードによる動作を体験しながら，Verilog HDLによるFPGA設計手法を学びます．具体的には，さまざまな簡単な回路をVerilog HDLで設計し，それらを組み合わせることによって小型CPUを実現します．最終的には，C言語のような高級言語で記述したプログラムを，設計した小型CPU上で動作させます．

設計した回路データは，実際にFPGAボードにダウンロードして動作を確認します（写真1）．ここでは米国Xilinx社の「Spartan-3Eスタータ・キット」を利用します．このスタータ・キットには，有効ゲート規模が約50万ゲートの「XC3S500E」が搭載されています．

FPGAベンダが提供している無料の設計ツールを用いて，シミュレーションによる動作確認も行うので，FPGAボードを持っていなくても学習には差し支えありません．ただし，FPGAボードで動作確認した方が，FPGA設計の楽しさをより実感できるでしょう．

● FPGAをVerilog HDLで設計する

Verilog HDLは，HDL（hardware description language；ハードウェア記述言語）と呼ばれる，ハードウェアを設計するための言語の一種です．ゲート・レベル（フリップフロップやANDゲートなどの論理回路の接続の記述）から，ビヘイビア・レベル（ハードウェアの動作やアルゴリズムの記述）までの，さまざまな抽象度で回路を設計することができます．記述方法はC言語のそれと似ており，回路図で設計するのが困難だった大規模な回路を，設計しやすくなります．

Verilog HDLと並んで有名なHDLとして，VHDLがあります．VHDLもVerilog HDLと同様に，さまざまな抽象度で回路を設計できます．VHDLはVerilog HDLと比べて，文法が厳格で記述量が多くなりやすい特徴があります．つまり，同じ動作をする回路をVHDLとVerilog HDLで設計すると，多くの場合はVerilog HDLの方が簡潔に記

写真1　Spartan-3Eスタータ・キット
Spartan-3Eスタータ・キットのボードには，50万ゲート相当のFPGA「XC3S500E-4FG320C」が搭載されている．各種メモリやインターフェースも充実しており，FPGA設計の入門者に適したキットである．

KeyWord　Verilog HDL，FPGA，HDL，全加算器，モジュール，ポート，シミュレーション，テストベンチ

述できます．しかし，文法が厳格な方が設計ミスが起こりにくいとも言えるので，Verilog HDLとVHDLのどちらが優れていると決めることはできません．本連載では，記述量が少なくて済み，初心者にとっつきやすいことから，Verilog HDLを選択しました．

● FPGA設計の基本的な流れ

Verilog HDLによるFPGA設計の基本的な流れを以下に示します（図1）．

1）最初に，Verilog HDLを用いて回路を作成します（これを「デザイン入力」と呼ぶ）．
2）次に，作成した回路が正しく動作するかをシミュレーションによって確認します．意図したように動作しない場合は，デザイン入力に戻って回路を修正し，再びシミュレーションを行って，正しく動作するまで繰り返します．
3）シミュレーションで正しく動作することが確認できたら，Verilog HDLで記述された回路記述を，ネット・リスト（基本的な回路から構成された回路記述）に変換します．これを「論理合成」と呼びます．
4）次に，使用するFPGAに合わせて，作成した回路の配置配線を行います．

図1
Verilog HDLによる回路設計の流れ
HDLで回路を記述し，シミュレーションで動作を確認してから論理合成を行う．その後，使用するFPGAに合わせて配置配線を行い，FPGAにダウンロードするビット・ファイルを作成する．

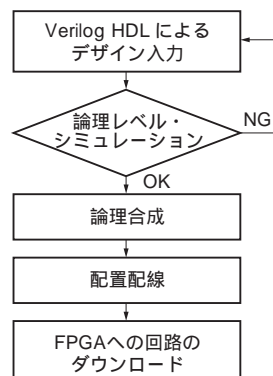



表1
全加算器の真理値表

入　力			出　力	
a	b	cin	s	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

5）配置配線が完了したら，ビット・ファイル（FPGAにダウンロードする回路データ）を作成します．最後に，作成したビット・ファイルをFPGAにダウンロードします．

実は，以上の作業は，FPGAベンダの提供するFPGA開発ツールを用いて行うことができます（特に，米国Xilinx社の「ISE WebPACK」，米国Altera社の「Quartus II Web Edition」などは無償で提供されている）．これらのツールは，Verilog HDL記述を基にして自動的に論理合成と配置配線を行い，ビット・ファイルを生成します．そのため，ユーザがネット・リストを見る必要はありません．

1 全加算器を設計してみる

それでは，実際にVerilog HDLを用いて回路を記述してみましょう．今回は全加算器（full adder）を取り上げます（図2）．

● まずは入出力の確認から

全加算器の入力はa，b，cinの3ビットで，出力はsとcoutの2ビットです．入力3ビットの和は，2ビットの2進数で表すことができます（“00”，“01”，“10”，“11”の4通り）．全加算器は，その上位ビットをcoutに，下位ビットをsに出力します．全加算器の入出力の対応を表した真理値表を表1に示します．

3ビットの入力a，b，cinのうち，奇数個が‘1’である場合に，出力sは‘1’となります．また，coutが‘1’となるのは，これらの3ビットのうち2ビット以上が‘1’の場合です．よって，論理式で書くと，

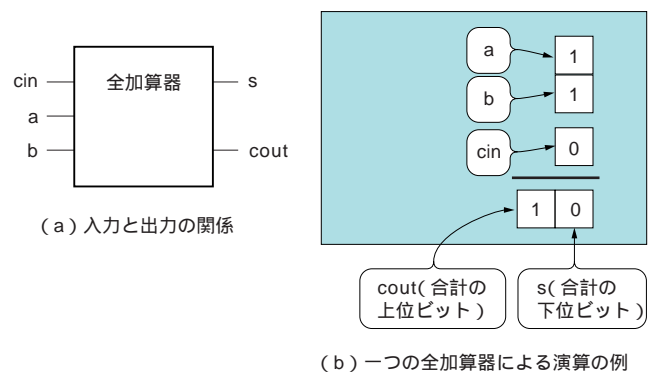



図2　全加算器
二つの値（a，b）を加算し，けた上がり入力（cin；carry in）を含めて演算する．結果として，出力値（s；sum）とけた上がり出力（cout；carry out）を出力する．全加算器を複数個用いることにより，複数ビットの加算に対応した加算器を作ることができるが，この設計方法については次回に詳しく説明する．

$$cout = (a \wedge b) \vee (b \wedge cin) \vee (cin \wedge a) \quad (1)$$

$$s = a \oplus b \oplus cin \quad (2)$$

となります(式の意味については，コラム1「全加算器の論理式を読み解く」を参照)．この論理式をもとに，全加算器をVerilog HDLで記述したのが**リスト1**です．

● Verilog HDL記述の解説

Verilog HDL記述は，「モジュール」と呼ばれる，入力ポートと出力ポートを持つひとまとまりの回路で構成します．モジュールは，モジュール宣言module(**リスト1**の1行目)で始まり，endmodule(**リスト1**の10行目)で終わります．

1行目のfaはモジュールの名前(モジュール名)がfaであることを示しています．通常，Verilog HDL記述は，モジュール名に拡張子vを付けたファイル名(この例では，fa.v)に保存します．

モジュール名の後の丸かっこの中は，そのモジュールのポート・リストです．ここでは，a，b，cin，s，coutのそれぞれが，入力または出力のためのポートであることを宣言しています．ポートとは，モジュール外部と値のやりとりを行うための出入り口のようなものです(**図3**)．モジュールはC言語の関数と似ており，ポートは関数呼び出しの引き数と同様の性質を持ちます．

3行目と4行目はポート宣言です．3行目のinputとそれに続くa，b，cinは，これらの三つのポートがそれぞれ1ビットの入力ポートであることを意味しています．また，4行目のoutputとそれに続くs，coutは，これら二つのポートが1ビットの出力ポートであることを宣言しています．

5行目のwireとそれに続くa，b，cin，s，coutは，これらがモジュール内部のネット(信号線)であることを宣言しています(ネット宣言)．ネットa，b，cinは入力ポートでもあるので，入力ポートの値がそのまま同じ名前のネットの値となります．つまり，入力ポートaとネットaは接続されているとみなされます．

同様に，ネットsとcoutは出力ポートでもあるので，ネットの値がそのまま出力ポートの値となります．ネットが入力ポートまたは出力ポートでもある場合，wire文によるネット宣言(5行目)は省略することができます．今後，このような場合は省略することにします．

7行目と8行目のassign文は，等号(=)の右辺にある式の評価値が，左辺のネットに継続的に書き込まれ続けることを意味します．つまり，右辺の式の値が変わると，直ちに左辺のネットに新しい値が書き込まれます．&，|，^は

リスト1　assign文を用いた全加算器のVerilog HDL記述(fa.v)

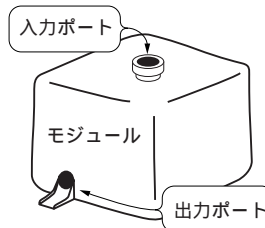
```
1  module fa(a, b, cin, s, cout);
2
3     input a, b, cin;
4     output s, cout;
5     wire a, b, cin, s, cout;
6
7     assign s = a ^ b ^ cin;
8     assign cout = (a & b) | (b & cin) | (cin & a);
9
10 endmodule
```


図3　モジュールとポート

ポートとは，モジュール外部と値のやりとりを行うための出入り口のようなものである．

表2　Verilog HDLのビット演算子

記号	説　明
~	NOT(ビット反転)
&	AND(ビット積)
|	OR(ビット和)
^	XOR(ビット排他的論理和)
~^	XNOR(ビット排他的否定論理和)

コラム1

全加算器の論理式を読み解く

本文中に示した論理式の意味を説明します．まず式(1)についてですが，coutは，(a∧b)，(b∧cin)，(cin∧a)という三つの項を論理和∨(いわゆる「または」)で結合したものです．つまり，三つの項のどれか一つでも‘1’になるとき，coutは‘1’になります．三つの項はそれぞれ3ビットのうち2ビットを取り出して組み合わせ，論理積∧(いわゆる「かつ」)で結合しているので，2ビット以上が‘1’のときにcoutが‘1’になることが分かります．

式(2)のsは，a，b，cinを排他的論理和⊕(相当する簡潔な日本語はないが，あえて言うと「どちらか一方のみ」)で結合しています．a⊕bは，aとbのいずれかが‘1’のときのみ‘1’になります．その結果，排他的論理和で結合された複数のビットは，‘1’が奇数個のときに演算結果が‘1’になります．

これにより，この二つの論理式が全加算器を正しく定めていることが分かります．

Verilog HDLのビット演算子です(**表2**)．ここでは，ネットsとcoutに適切な値が継続的に代入されます．このようにassign文を用いることにより，信号線の永続的な接続関係を定義することができます．

2 設計ツールを使用してVerilog HDLを記述する

それでは，FPGAで回路設計を行うためのツール「ISE WebPACK」を使って，**リスト1**を実際に入力してみましょう．ISE WebPACKの入手方法やインストールについては，コラム2「ISE WebPACKのインストール」を参照してください．

● プロジェクトの作成

ISE WebPACKを起動して，プロジェクトを作成します．プロジェクトとは，回路設計に必要なソース・ファイルや設定ファイルをまとめたものであり，設計を始めるときに最初に作成する必要があります．

まず，メニュー・バーから「File」→「New Project...」を選択します．ウィザード形式のウィンドウが表示されるので，1ページ目のCreate New Projectウィンドウでは，プロジェクト名，作業ディレクトリ，最上位回路のソース・タイプを設定します[**図4(a)**]．ここでは，プロジェクト

コラム2

Column ISE WebPACKのインストール

ISE WebPACKは，Xilinx社より無償で提供されているFPGA設計ツールで，Xilinx社のWebサイト(http://japan.xilinx.com/)からダウンロードできます(ただしユーザ登録が必要)．2007年2月時点では，ISE WebPACKのバージョン9.1iがダウンロード可能です注A．

● ISE WebPACKのインストール

まず，ISE WebPACKをインストールします．インストール時にインストール先のフォルダを指定することができますが，デフォルト(c:\Xilinx)のままにするのがよいでしょう．ISE WebPACKを含め，一般の設計ツールでは，フォルダ名(インストール先や設計データの格納フォルダ)にスペースや日本語が含まれると，正しく動作しないことがあるので，気を付けてください．

● ISE WebPACKを起動する

インストールが終わったら，スタート・メニューから「プログラム」-「Xilinx ISE 9.1i」-「Project Navigator」を選択し，ISE WebPACKを起動します(**図A**)．

Project Navigatorは，Sourcesウィンドウ，Processesウィンドウ，ワーク・スペース，Transcriptウィンドウによって構成されています．各ウィンドウは機能ごとにタブが付いており，用途に応じてそれを切り替えて使用します．

各ウィンドウの説明を以下に示します．

- Sourcesウィンドウ
  プロジェクト(後述)に関する操作を行います．「Sources」タブ，「Snapshot」タブ，「Libraries」タブから構成されます．
- Processesウィンドウ
  Sourcesウィンドウで選択されたファイルに対して，プロセスを実行します．表示される実行可能なプロセスは，選択するソース・ファイルのタイプによって異なります．
- ワーク・スペース
  ソース・ファイルの編集や，シミュレーション波形の表示を行います．複数のソース・ファイルやシミュレーション波形の表示が可能で，タブをクリックすることによって表示内容を切り替えることができます．
- Transcriptウィンドウ
  実行したプロセスの出力メッセージが表示されます．ErrorsタブやWarningsタブなどをクリックすることで，表示されるメッセージを絞り込むことができます．

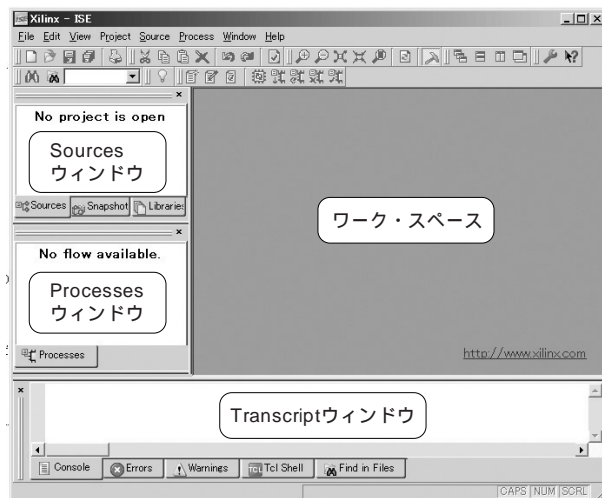
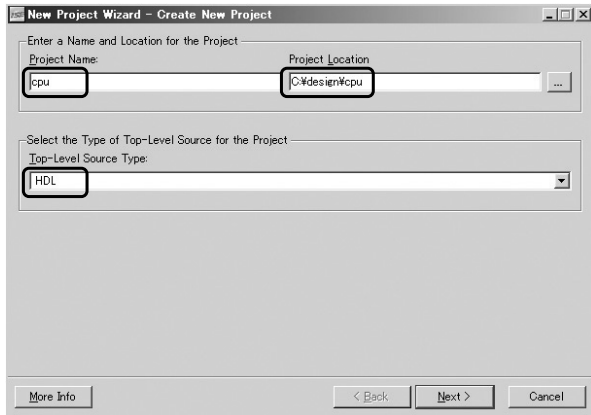
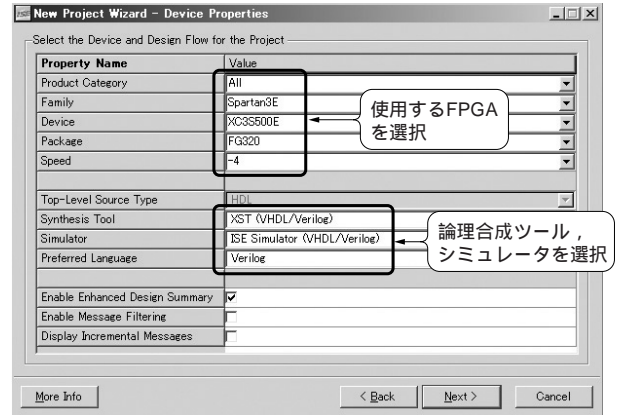
注A：Xilinx社の販売代理店からも，無償でISE WebPACKのDVD-ROMを入手できる．また，本誌2007年3月号の付属DVD-ROMには，ISE WebPACK 8.2iを収録している．

図A　ISE WebPACKの起動画面
このツールで，デザイン入力からFPGAへのダウンロードまでのすべての作業を行うことができる．

(a) プロジェクト名と作業ディレクトリの設定

(b) 使用するデバイスの設定

図4　新規プロジェクトの作成

名を「cpu」，作業ディレクトリを「c:\design\cpu」，ソース・タイプを「HDL」に設定します．プロジェクト名や作業用ディレクトリには，スペースや日本語が含まれないようにした方がよいでしょう．[Next] をクリックして2ページ目に移ります．

2ページ目のDevice Propertiesウィンドウでは，使用するデバイスを設定します[**図4(b)**]．ここでは，Spartan-3E スタータ・キットに合わせて，Familyは「Spartan3E」，Deviceは「XC3S500E」，Packageは「FG320」，Speedは「－4」をリストから選択し，残りの項目はデフォルト(最初の状態)のままにします．念のため，Synthesis Toolは「XST(VHDL/Verilog)」，Simulatorは「ISE Simulator (VHDL/Verilog)」であることを確認してください．本連載では，これらの論理合成ツールとシミュレータを使います．設定できたら，[Next] ボタンをクリックします．

3ページ目のAdd Existing Sourcesウィンドウでは，既存のソース・ファイルをプロジェクトへ追加できます．ここでは追加しないので [Next] をクリックし，次のページに移ります．

最後のProject Summaryページで，作成するプロジェクトの概要が表示されます．[Finish] をクリックすれば，プロジェクトの作成が完了します．

● ソース・ファイルの作成

次に，ソース・ファイルを作成します．まず，Sourcesウィンドウ内を右クリックし，「New Source...」を選択します．ウィザードのウィンドウが表示されるので「Verilog Module」を選択し，File nameに「fa」と入力して，[Next] ボタンをクリックします(**図5**)．

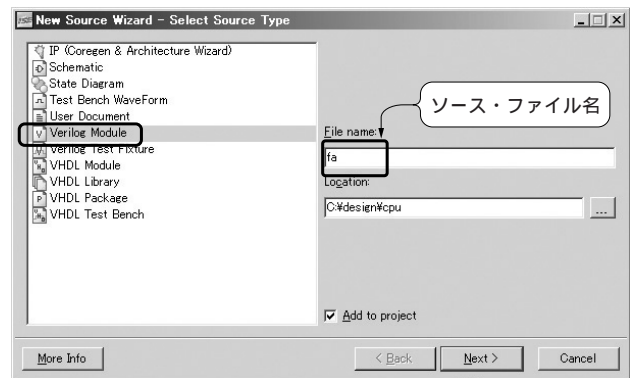



図5　新規ファイルの作成

次のページでは，入出力ポートを設定します．ここでは直接ソース・ファイルに記述するので，何も入力せずに [Next] ボタンをクリックして，次のページの [Finish] ボタンをクリックすると，ソース・ファイル(fa.v)が生成されます．Sourcesウィンドウのリストにfa(fa.v)が追加されたのを確認してください．また，ワーク・スペースにfa.vのテンプレートが表示されます．このテンプレートを編集することにより，全加算器を設計します．

それでは，fa.vのテンプレートを編集して，**リスト1**のように書き換えてみてください(左の行番号は，リストを見やすくするために付加しているものなので，入力しない)．編集が完了したら，メニュー・バーから「File」→「Save」を選択して，ファイルを保存します．

● 回路記述の構文チェック

次に，構文の誤りやスペル・ミスがないかどうかを確認しましょう．Sourcesウィンドウ上部のドロップダウン・リストから「Synthesis/Implementation」を選択して，「fa (fa.v)」をマウスで選択(反転)します．Processesウィンドウの「Synthesize - XST」の階層を展開し，「Check Syntax」をダブルクリックします(**図6**)．構文に誤りがある場合は，その旨がTranscriptウィンドウに表示されるので，誤りを修正します．

● 回路図の確認

構文チェックで記述の誤りがないことを確認したら，設計した回路の構成を回路図で確認してみます．まず，Processesウィンドウの「Synthesize - XST」の階層を展開し，「View RTL Schematic」をダブルクリックします(**図7**)．すると，ワーク・スペースに入力ポートと出力ポートが並んだ図が表示されます[**図8(a)**]．これは，作成したモジュールの入出力ポートを示したものです．この四角形の内側をダブルクリックすることで，モジュール内部の回路構成を見ることができます[**図8(b)**]．ここに示されているXOR3は，a，b，cinを入力，sを出力として持つ，3入力排他的論理和ゲートです．この論理ゲートは，**リスト**

図6　構文チェック

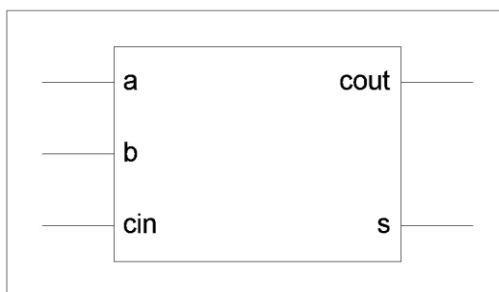
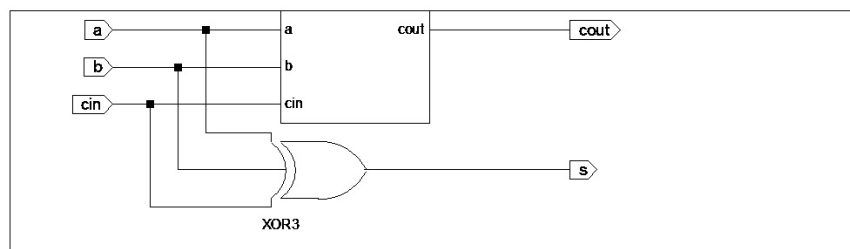
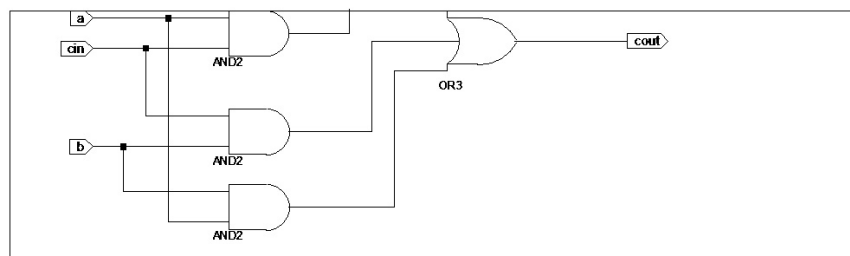
図7　回路図の確認

(**a**) 全体像

(**b**) (**a**) のブロック(四角形)をブレーク・ダウンした

(**c**) (**b**) のブロック(四角形)をブレークダウンした

図8　作成した回路図

1の7行目の記述に対応しています．また，その上にある四角形は8行目の記述に対応しており，そこをダブルクリックすることで，その中の構成を見ることができます[**図8(c)**]．このとき，一つ前の図に戻るためには，図の外側の何もない部分をダブルクリックします．

このように，Verilog HDLで設計した回路を回路図で見ることができます．Verilog HDLの記述に慣れないうちは，自分の設計した回路がどのような回路図になるのかを確認しながら進めた方がよいでしょう．上級者になると，Verilog HDL記述がどのような回路を生成するかを念頭に置きながらハードウェア設計を行い，コンパクトで高速な，効率の良いハードウェアを設計できるようになります．

3 シミュレーションによる動作確認

続いて，作成した回路の動作を確認するために，シミュレーションを行います(入力ポートに対して外部から信号を時系列で与え，出力ポートの信号を観測する)．シミュレーションを行うには，入力信号について時系列で記述したテスト・ベンチを作成する必要があります．テスト・ベンチは回路と同様に，Verilog HDLを用いて記述します．

● テスト・ベンチの作成

テスト・ベンチを作成する手順を説明します．Sourcesウィンドウ内を右クリックして「New Source」を選択すると，新規ソース・ファイル作成ウィザードが表示されます．ここで，fa.vを作成したとき(**図6**)と同様に「Verilog Module」を選択し，File nameを「fa_tb」として，fa_tb.vを作成します．Sourcesウィンドウにfa_tb.vのテンプレートが表示されるので，これを編集してテスト・ベンチを作成します．**リスト2**に，全加算器回路(fa.v)のテスト・ベンチの例を示します．

1行目の`` `timescale 1ns / 1ps``は，シミュレーションの時間設定をしています．書式は次の通りです．

`` `timescale [単位時間] / [精度]``

リスト2の記述の場合は，「1単位時間を1nsとし，精度1psでシミュレーションを行う」という意味になります．

テスト・ベンチの記述も，回路記述と同様にモジュールから構成されています．モジュールは，2行目のモジュール宣言`module`で始まり，19行目`endmodule`で終わります．ただし，テスト・ベンチはポートを持ちません．

4行目の`reg`はレジスタ宣言です．動作を確認するモジュールの入力ポートに接続する信号線`a`，`b`，`cin`を，レジスタ宣言を用いて，レジスタ型変数として定義しています．回路記述のところで用いた継続的な代入の場合はネット宣言を用いましたが，これらの信号線は後述の`initial`文で時間経過によって値を変化させるので，レジスタ宣言を使用します．ネット宣言とレジスタ宣言の使い分けは，次回に詳しく説明します．

5行目は，シミュレーションを行う回路の出力ポートに接続するネット`s`，`cout`を宣言しています．

6行目は，シミュレーションを行う回路のインスタンス宣言をしています．インスタンス宣言とは，他で定義されたモジュールをインスタンス化(実体化)するもので，C言語の関数呼び出しと似ています．インスタンス宣言の書式は次の通りです．

```
[モジュール名] [インスタンス名](.[ポート名]
([接続信号名]), ...);
```

ここでは，先ほど作成した全加算器のモジュール名(`fa`)と，インスタンス名として`fa0`を記述し，それぞれの入出力ポートとその接続信号の関係を列挙しています．

8行目～18行目は`initial`文であり，入力信号の値を設定します．このうち9行目で，入力信号の初期値を設定します．次に，10行目で100単位時間，ここでは100ns経過した後の入力信号の値を設定します．このように，`#[単位時間]`でシミュレーションの時間経過を表します．これを繰り返すことによって，時間経過における各入力信号の値を設定します．

リスト2のテスト・ベンチを入力したら，メニュー・バーから「File」→「Save」を選択して，忘れずに保存しましょう．

リスト2　テストベンチ(fa_tb.v)

```
 1 `timescale 1ns / 1ps
 2 module fa_tb;
 3
 4   reg a, b, cin;       ← レジスタ宣言
 5   wire s, cout;
 6   fa fa0 (.a(a), .b(b), .cin(cin), .s(s), .cout(cout));  ← インスタンス宣言
 7
 8   initial begin
 9     a = 0; b = 0; cin = 0;        ┐
10     #100 a = 1; b = 0; cin = 0;   │
11     #100 a = 0; b = 1; cin = 0;   │
12     #100 a = 1; b = 1; cin = 0;   │
13     #100 a = 0; b = 0; cin = 1;   ├ ← initial文
14     #100 a = 1; b = 0; cin = 1;   │
15     #100 a = 0; b = 1; cin = 1;   │
16     #100 a = 1; b = 1; cin = 1;   │
17     #100 a = 0; b = 0; cin = 0;   ┘
18   end
19 endmodule
```

● テスト・ベンチのチェックと実行

次に，回路と同様に構文チェックを行います．まず，Sourcesウィンドウ上部のドロップダウン・リストから「Behavioral Simulation」を選択します．次に，Sourcesウィンドウのfa_tb(fa_tb.v)を選択し，Processesウィンドウの「Xilinx ISE Simulator」の階層を展開して，「Check Syntax」をダブルクリックします(**図9**)．

次に，Processesウィンドウの「Simulate Behavioral Model」をダブルクリックすると，ワーク・スペースにシミュレーション結果が表示されます(**図10**)．このとき，左の信号名を上下にドラッグすることによって，各信号の位置を入れ替えることができます．

シミュレーション結果が，**表1**の値と同じであることを確認してください．もし結果が異なっていたら，Verilog HDL記述に誤りがあるはずなので，ソース・ファイルを見直します．

シミュレーション実行後，Processesウィンドウに「Sim Hierarchy」タブが追加されます．このタブで階層を展開すると，回路の各信号のリストが表示されます．その信号を波形ウィンドウにドラッグすれば波形ウィンドウに追加でき，再びシミュレーションを行うことによって，その信号の値を調べることができます．

以降，Processesウィンドウでは，ファイルに対してプロセスを実行したい場合は「Processes」タブに，シミュレーション波形に対して操作したい場合は「Sim Hierarchy」タブに切り替えて作業を行います．

4 FPGAボードを用いた動作確認

シミュレーション結果が正しいことを確認したら，実際にFPGAで動作させてみましょう．

● ピンを割り当てる

まず，Verilog HDL記述にあるポートとFPGAのピンの対応付け(ピン割り当て)を行います．ピン割り当てはユーザ制約ファイル(UCFファイル)に記述します．ここでは，入力ポートをボード上のスイッチに，出力ポートをLEDに接続するようにして，スイッチの状態によってLEDの点灯状態が変化するようにしてみましょう．

まず，Sourcesウィンドウ上部のドロップダウン・リストから「Synthesis/Implementation」を選択します．次に，Sourcesウィンドウの「fa(fa.v)」を右クリックして「New Source...」を選択します．

ウィザードが表示されるので，「Implementation Con straints File」を選択し，File nameに「fa」と入力して，[Next]ボタンをクリックします．次のページで[Finish]ボタンをクリックして，UCFファイル(fa.ucf)を作成します．

次に，Sourcesウィンドウの「fa.ucf(fa.ucf)」を選択しま

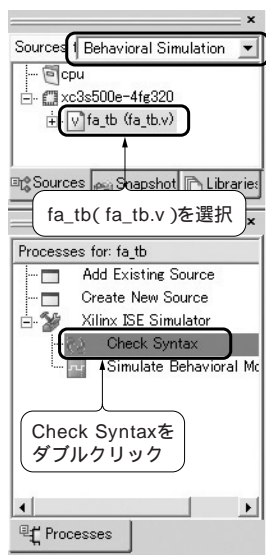



図9　テストベンチの構文チェック

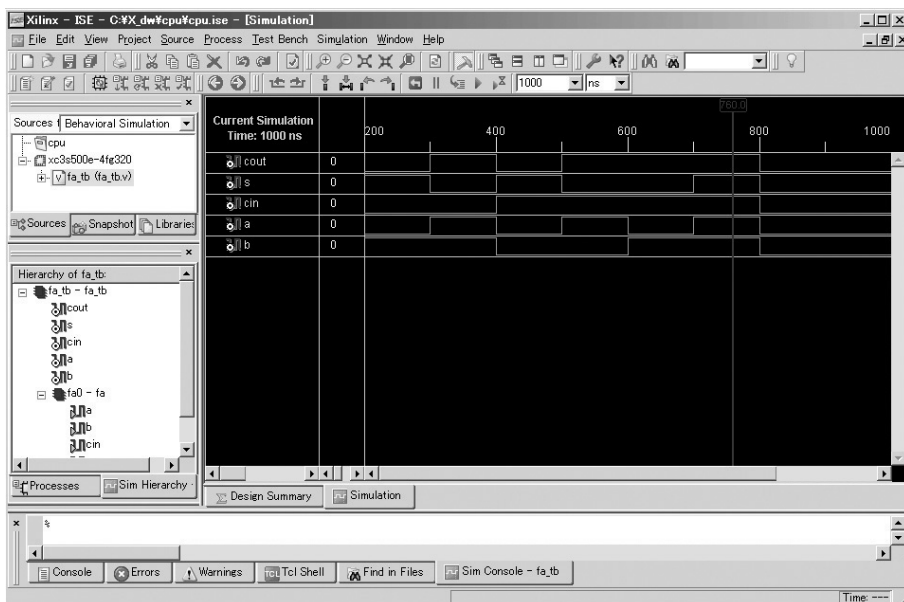



図10　シミュレーション波形

す．Processesウィンドウの「Processes」タブをクリックし，「User Constraints」の階層を展開します．そして，「Edit Constraints (Text)」をダブルクリックします．空白のファイルがワーク・スペースに表示されるので，**リスト3**を入力します．

UCFファイルでは，各行の#以降はコメントとして扱われます．また，NETから始まる文で，モジュールの各ポートの設定を記述します．fa.vの場合，五つのポートa，b，cin，s，coutに対する設定を記述します．書式は次の通りです．

```
NET "[ポート名]" [制約条件] (| [制約条件]...);
```

各ポートに対して，LOCはポートの接続するピンの指定，IOSTANDARDは入出力標準の割り当て，PULLUPはポートのプルアップ設定，SLEWは出力バッファのスルーレートの指定，DRIVEは信号強度の割り当てを行います．

例えば，2行目では入力ポートaをL13ピンに接続し，入出力標準をLVTTLに，ポートをPULLUPに設定しています．ピンL13，L14，H18はFPGAボードのスライド・スイッチと，F12とE12はLEDと接続しています．

詳しくはボードのマニュアル(1)を参照してください．

● ビット・ファイルのダウンロード

次に，作成した回路をボード上のFPGAにダウンロードする方法について説明します．具体的には，回路記述に基づいて拡張子が.bitであるビット・ファイルを作成し，FPGAにダウンロードします．

まず，スタートアップ・クロックを設定します．Sourcesウィンドウ上部のドロップダウン・リストから「Synthesis/Implementation」を選択してfa (fa.v)を選択し，Processesウィンドウの「Generate Programming File」を右クリックして「Properties...」を選択します．「Process Properties」ウィンドウが表示されるので，Categoryから「Startup Options」を選択し，FPGA Start-Up Clockを「JTAG Clock」に設定し，［OK］ボタンをクリックします（**図11**）．

次に，Processesウィンドウの「Generate Programming File」をダブルクリックします．エラーがなければ，fa.bitというビット・ストリーム・ファイルが生成されます．このファイルをFPGAにダウンロードすることによって，FPGAの中身を書き換えることができます．

次に，FPGAボードとパソコンをUSBケーブルで接続します．ボードの電源を入れると，自動的にUSBデバイスとして認識されます．

初めて接続する場合は，デバイス・ドライバをインストールする必要があります．もっとも，ISE WebPACKにデバイス・ドライバが含まれており，ISE WebPACKをインストールしたときにデバイス・ドライバもインストールされています．よって，接続したときに表示されるハードウェア・ウィザードで「ソフトウェアを自動的にインストールする（推奨）」を選択することにより，インストールできます．この操作は一度だけ行えばよく，次回接続した際には表示されません．

次に，Processesウィンドウの「Generate Programming File」の階層を展開し，「Configure Device (iMPACT)」をダブルクリックします．Welcome to iMPACTウィンドウが表示されるので，「Configure device ...」を選択し，リストから「Automatically ...」を選択して，［Finish］ボタンをクリックします．FPGAボードが正しく接続されていれば，Assign New Configuration Fileウィンドウが表示さ

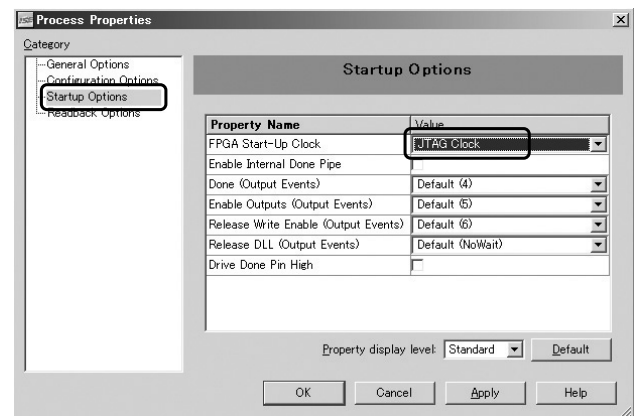



図11　スタートアップ・クロックの設定

リスト3
ユーザ制約ファイル (fa.ucf)

```
1   # SWITCH
2   NET "a" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP;
3   NET "b" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP;
4   NET "cin" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP;
5
6   # LED
7   NET "s" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8;
8   NET "cout" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8;
```

図12
回路データをダウンロードするツール「iMPACT」

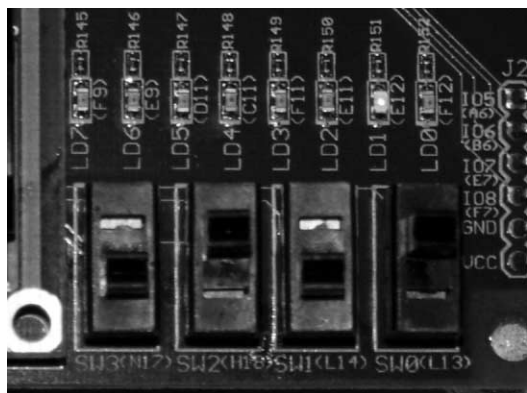
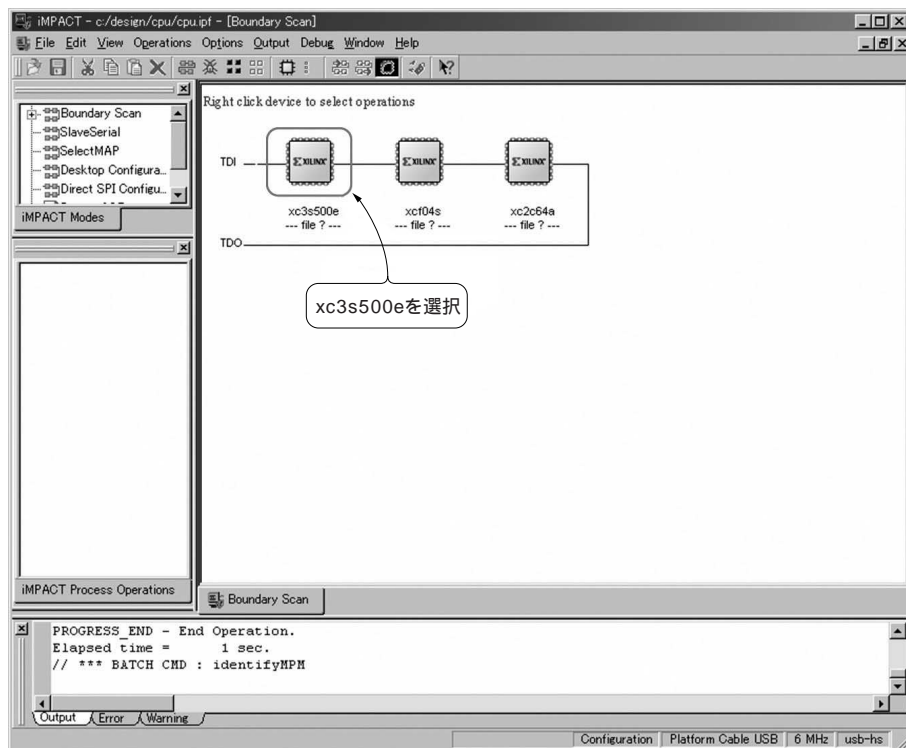



写真2　スライド・スイッチとLED
スライド・スイッチを切り替えると，LEDの状態(点灯/消灯)が変化する．

れます．後でファイルの指定をするので，ここではすべてのウィンドウに対して「Cancel」ボタンをクリックします．すると，**図12**の画面が表示されます．

次に，「xc3s500e」と表示されている上にあるFPGAのアイコンを右クリックして，「Assign New Configuration File...」を選択します．先ほど作成した「fa.bit」を選択し，[Open]ボタンをクリックします．「xc3s500e」と表示されている図を右クリックして「Program...」を選択すると，Programming Propertiesウィンドウが表示されるので，[OK]ボタンをクリックします．すると，FPGAに回路データがダウンロードされます．

スライド・スイッチの右側から順に，入力ポートa，b，cin，LEDの右側から順に出力ポートs，coutに対応しています(**写真2**)．スイッチのON/OFFを切り替えてみて，実際にLEDが正しく点灯することを確認しましょう．

次回は，全加算器を複数個用いた4ビット加算器や，より複雑な算術論理演算回路を設計します．

参考・引用*文献

(1) Spartan-3E スタータ キット ボード ユーザー ガイド(日本語版)，http://direct.xilinx.com/bvdocs/userguides/j_ug230.pdf

なかの・こうじ
いとう・やすあき
広島大学大学院 工学研究科

<筆者プロフィール>
中野浩嗣．1992年，大阪大学大学院 博士後期課程修了．工学博士．一つの民間企業，二つの大学を経て，2003年より広島大学教授．
伊藤靖朗．2003年，北陸先端科学技術大学院大学 博士前期課程修了．現在，広島大学助手．